

**REMARKS**

The claims and drawings have been amended to correct typographical errors and to clarify the scope of the claims. No amendments have been made for reasons relating to patentability. The Specification and Abstract have been amended to expedite examination. No new matter has been introduced by way of this amendment. Full examination and favorable action are requested.

Applicant believes the fee for filing these papers is \$18.00. A check in that amount is enclosed. If, however, Applicant is in error the Commissioner is hereby authorized to charge the deposit account of Rosenthal & Osha L.L.P., Deposit Account No. 50-0591 (Reference No. 16159/005001), for any additional fees and/or credits during the entire pendency of this application. A duplicate copy of this paper is enclosed for accounting purposes.

Respectfully submitted,

Date: 12/3/01

  
Jonathan P. Osha, Reg. No. 33,986  
**ROSENTHAL & OSHA L.L.P.**  
700 Louisiana, Suite 4550  
Houston, TX 77002

Telephone: (713) 228-8600  
Facsimile: (713) 228-8778

**APPENDIX A – MARKED-UP VERSION OF THE AMENDED CLAIMS**

**[c26]** (Amended) A method for testing and debugging a web application, comprising:  
sending a requested execution mode to the debugging controller;  
determining an execution mode of a server;  
comparing the requested execution mode and the execution mode of the server;  
switching the execution mode of the server to match the requested execution mode;  
**[and]**  
forwarding the HTTP request to the execution server[.]; and  
running and modifying each reply request with a portion of the collected data  
associated with the portion of a prior HTTP request.

**[c30]** (Amended) A tool for testing and debugging a web application, comprising:  
means for sending a requested execution mode to the debugging controller;  
means for determining an execution mode of a server;  
means for comparing the requested execution mode and the execution mode of the  
server;  
means for switching the execution mode of the server to match the requested  
execution mode; **[and]**  
means for forwarding the HTTP request to the execution server[.]; and  
means for running and modifying each reply request with a portion of the collected  
data associated with the portion of a prior HTTP request.

**APPENDIX B – MARKED-UP VERSION OF THE SPECIFICATION**

[0046] (Amended) In general, in one aspect, the invention comprises a method for testing and debugging a web application. The method comprises sending a requested execution mode to the debugging controller, determining an execution mode of a server, comparing the requested execution mode and the execution mode of the server, switching the execution mode of the server to match the requested execution mode, **[and]** forwarding the HTTP request to the execution server[.], and running and modifying each reply request with a portion of the collected data associated with the portion of a prior HTTP request.

[0047] (Amended) In general, in one aspect, the invention comprises a tool for testing and debugging a web application. The tool comprises means for sending a requested execution mode to the debugging controller, means for determining an execution mode of a server, means for comparing the requested execution mode and the execution mode of the server, means for switching the execution mode of the server to match the requested execution mode, **[and]** means for forwarding the HTTP request to the execution server[.], and means for running and modifying each reply request with a portion of the collected data associated with the portion of a prior HTTP request.

[0056] (Amended) The present invention allows the user to transparently replay a HTTP request while simultaneously **[restarting the server in]**switching the server to debugging mode and opening the debugging client. Once this occurs the error can be readily identified through source level debugging using the debugging client. Once the error has been identified and fixed, the present invention allows the user to transparently switch back to normal execution mode. In normal execution mode, the present invention allows the user to replay the HTTP request to verify that the error was fixed.

[0063] (Amended) The client-side component 18 also includes a debugging controller 19 that controls the execution mode of the server *i.e.*, normal or debugging, and the mechanism for allowing transparent switching between the two modes. The

normal mode corresponds to regular execution of the execution server 10. The debugging mode corresponds to executing the HTTP requests with a debugging session on the execution server 10. When the client-side component issues a replay request, it first contacts the debugging controller 19 to check whether the execution server 10 which the request is intended to run on is in the desired mode. If not, [it] the client-side component 18 issues a request to the debugging controller 19 to switching the mode of**[restart]** the execution server 10. Once the client-side component 18 receives confirmation from the debugging controller 19 that the server is running in the desired mode, it issues the request**[through an internal HTTP server 12]**.

[0073] (Amended) The “Replay” action invokes the debugging controller 19 to detect if the execution server 10 upon which the HTTP request was originally executed on is running. If the execution server 10 is not running, the execution server 10 is started. The execution server 10 is started in the normal mode if button 35 is toggled. The execution server 10 is started in the debugging mode if the button 33 is toggled. If the execution server 10 is running in normal mode and button 33 is toggled, the execution server 10 is switching to**[restarted in]** debugging mode. If the execution server 10 is running in debugging mode and button 35 is toggled, the execution server 10 is switched to**[restarted in]** normal mode. The HTTP request is then sent to the execution server 10.

[0074] (Amended) The “Edit and Replay” action brings up a dialog, which allows the user to modify the HTTP request data prior to causing it to be re-processed by the execution server 10. Additionally, the user can modify which execution server 10 the user wants to execute on through the dialog. When the user completes the editing of the HTTP request and sends the HTTP request to the execution server 10, the debugging controller 19 intercepts and examines the HTTP request. If the execution server 10 upon which the HTTP request is to be run is not running, the execution server 10 is started. The execution server 10 is started in the normal mode if button 35 is toggled. The execution server 10 is started in the debugging mode if the button 33 is toggled. If the execution server 10 is running in normal

mode and button 33 is toggled, the execution server 10 is switched to~~[restarted in]~~ debugging mode. If the execution server 10 is running in debugging mode and button 35 is toggled, the execution server 10 is switched to~~[restarted in]~~ normal mode. The modified HTTP request is then sent to the execution server 10.

[0076] (Amended) “Replay sequence” causes the HTTP requests described by each sequence folder to be replayed one at a time in sequence. With “Replay sequence (step),” the output from each request is shown before moving on to the next request. With “Replay sequence (to end),” only the output of the last request is shown. To execute the sequence, a thread is started which attempts to execute each request in order, waiting for one to succeed before executing the next one. Each individual request has have a timeout, which can be some default number or can be set by the user. If any individual request fails, the output of that request is displayed to the user, and the user is notified of the problem. Otherwise, the output of the last request in the sequence is displayed. For each replay request within the sequence the debugging controller 19 intercepts and examines the HTTP request. If the execution server 10, upon which the HTTP request is to be run is not running, the execution server 10 is started. The execution server 10 is started in the normal mode if button 35 is toggled. The execution server 10 is started in the debugging mode if the button 33 is toggled. If the execution server 10 is running in normal mode and button 33 is toggled, the execution server 10 is switched to~~[restarted in]~~ debugging mode. If the execution server 10 is running in debugging mode and button 35 is toggled, the execution server 10 is switched to~~[restarted in]~~ normal mode. The HTTP request is then sent to the execution server 10.

[0081] (Amended) The debugging controller 19, based on the aforementioned information, may send a command 57 to execution server 10, to start the execution server 10 in the mode specified in the GUI 22. Based on the command 57, the execution server 10~~[IDE 6]~~ may be exited and ~~[the execution server 10]~~ may be restarted. In one implementation, debugging, for portions of the execution server 10, may be activated without ~~[exiting the IDE 6 and]~~ shutting down the entire execution server 10. Once the mode is selected, the debugging controller 19 sends a

confirmation to the client-side component 16. The client-side 16 subsequently sends the request to the execution server 10 via an internal HTTP server 12.

[0084] (Amended) In one implementation, the debugging controller 19 uses a "Server Integration" API for controlling the server execution functionality. In this implementation, each execution server provides an implementation of the API called a server integration plugin. The server integration plugin uses proprietary methods for starting, stopping, and turning debugging on/off on the execution server. When the debugging controller receives a request to turn on debugging, the server integration plugin turns on debugging on the execution server using proprietary methods.

[0087] (Amended) To determine if the execution server 10 is in the proper operating mode, the debugging controller 19 first determines what mode the execution server 10 has requested (Step 104). The debugging controller 19 determines the requested mode by checking which button 33 or 35 (in Figure 2) is toggled. Next, the debugging controller 19 determines the current operating mode of the execution server 10 (Step 105). If the current execution mode and the requested mode are the same (Step 106), the debugging controller 19 forwards the HTTP request to the execution server 10 (step 108). If the current execution mode and the requested mode are not the same (Step 106), the debugging controller 19 switches to the requested mode (Step 107). The debugging controller 19 **[switches the execution mode, by exiting the execution server 10, turning on or off the debugger (depending on which mode is required), and then restarting the execution server 10.]**uses the server integration plugin to turn on or off the debugger (depending on the mode required) using proprietary methods. Once this has been completed the **[internal HTTP server 12]**IDE 6 forwards the HTTP request to the execution server (Step 108).

**APPENDIX C – MARKED-UP VERSION OF THE ABSTRACT**

A system for monitoring HTTP transactions between a server and a client, including a data collector which runs on the server and collects data from HTTP requests sent by the client to the server and data from HTTP responses sent by the server to the client,[;] a debugging controller which controls an execution mode of the server[. A], a graphical display which displays the collected data and through which replay requests are sent to the server, each replay request specifying a prior HTTP request to be interpreted by the server[. A], and a request player which runs on the server and modifies each replay request with a portion of the collected data associated with the prior HTTP request to be interpreted by the server.

22698\_1